Account Allocations on the Grid

Thomas J. Hacker
Center for Parallel Computing
University of Michigan

Brian D. Athey
Cell & Developmental Biology
University of Michigan

Email: {hacker,bleu}@umich.edu

Abstract

The Grid infrastructure provides access for a large pool of users to a large number of distributed computing resources. Providing access for the complete pool of potential users would put an unacceptably large administrative burden on sites that participate in the Grid. Current approaches to solve this problem require an account for each user at a site, or maps all users into one account. This paper proposes an alternative approach to account allocation that provides the benefits of persistent accounts while minimizing the administrative burden on Grid resource providers. A technique for calculating the upper bound on the number of jobs and users offered to the system from the Grid that is based on historical use is presented. Finally, application of this approach to the National Institutes of Health Visible Human Project is described.

1 Introduction and Motivation

When the Grid computing environment [29] becomes fully operational, potentially tens of thousands of users will be actively using resources made available to them by Grid resource providers. To access these resources, each of these users will require some form of a local account. However, requiring each participating site on the Grid to accurately maintain tens of thousands of accounts is neither desirable, nor feasible.

Many sites will likely have a finite set of users that regularly visit the site for resources. If a Grid user utilizes resources at a site infrequently, or only once, the overhead associated with creating, maintaining and deleting an account for the infrequent user would quickly overwhelm systems staff and provide a strong disincentive for resource sites to participate in the Grid.

To address this problem, a mechanism for provisioning "template" accounts for Grid users in a local environment is proposed, that provides a mechanism for local resource administrators to provide relatively instantaneous access to local resources, persistence for frequent users, and enables close tracking and accounting of Grid user resource utilization. To predict an upper bound on the number of individual jobs and unique users that will utilize the system, a technique based on historical job logs is presented in the last section of this paper.

The proposed mechanism replaces the fundamental systems paradigm of a strong binding between a real user and an account on a system with the paradigm of a temporary binding between an account and a real user. Authentication and authorization that traditionally has been performed on the local system is replaced with distributed authentication and authorization systems, such as Kerberos [5], X.509[6], and Akenti [2]. To enforce user accountability, local record keeping can be used to track the actions and resource utilization of users.

The impetus for this mechanism is expressed in the Grid Forum draft Security Implications of Typical Grid Computing Scenarios [1]: "For each resource that requires local user ids, the Grid system administrator on the resource must determine the mapping of Grid ID…to local user ID…into a Grid-specific mechanism…" The need for a simple and fast administration is expressed in [34].

Two large Grid projects under development will directly benefit from the account template mechanism described in this paper. One project is the system at Pacific Northwest National Laboratories [39] that is focused on real-time dynamic allocation and accounting mechanisms for computational resources.
The other large Grid project that will benefit is the Information Power Grid at NASA [30], which is attempting to build a production network of high performance computing resources and data storage devices.

2 Current Situation and Related Work

The problem of maintaining accounts across a large number of hosts in a local administrative domain has been addressed over the years with a number of solutions. Project Athena at MIT solved the problem in the UNIX workstation space by using Kerberos for verifying user identity, and Hesiod [4] for maintaining password files across a large number of workstations. This central control mechanism worked well for a small class of hosts (UNIX workstations) in a well-defined administrative domain.

As the class of desktop hosts that required centralized authentication and authorization mechanisms increased, attempts were made to extend existing solutions to cover the new systems. Kerberos authentication was added to Apple Macintosh [13] and Microsoft Windows platforms [14]. Other authentication and authorization systems were created that addressed the problem well in one host class, but poorly in others. For example, Novell addressed the problems in the Microsoft Windows space very well, but only marginally supported UNIX workstation hosts.

Carnegie Mellon University's Andrew Project [15] created a wide area filesystem primarily targeted to UNIX workstation class hosts. For authentication, the Andrew project adopted MIT Kerberos for local domain (cell) authentication, but never fully addressed the problems of cross-domain authentication. The authorization mechanisms in Andrew were designed to control file access, and were also centered on the local administrative domain.

The Distributed File System (DFS) [16] created a centralized authorization and authentication mechanism (the Registry) that attempted to address the local and cross-domain authentication and authorization issues. In practice, however, the effort required to maintain a large central repository of information became overwhelming, and the performance of the registry was poor [17].

Web based backup, collaboration and virtual office services, such as Egroups.com, and web based collaboration and file storage systems have emerged in the last few years that utilize X.509 based authentication to provide higher level distributed services.

Several attempts are now being made in the Gird computing milieu to address the cross-domain authentication and authorization problems. Globus [18] is utilizing X.509 based authentication mechanisms to successfully deploy a computational job across a set of supercomputer systems. Grid resource provider sites maintain a file that maps from X.509 distinguished names (DNs) into a local account identifier that was created for the individual user.

All of the systems described work well in finite local domains, or when the number of users is finite and less then a few thousand. However, when attempts are made to utilize resources across administrative domains or when the number of users exceeds 100,000, the solutions fail to scale elegantly.

Other Account Allocation Approaches

Condor solves this problem by using one UID ("nobody") to execute jobs for users that do not have an account in a Condor flock [10]. The PUNCH system uses a similar scheme, where all users are represented by logical user accounts within a single physical account [11, 12] with the ability to use a set of dynamic account bindings for system calls.

There are disadvantages to these approaches. If there are multiple jobs on the same system from different users, with all of the users assigned to one UID, it is difficult for the system to distinguish between those users since they all share the same UID. An example of this problem with identity is if a common scratch disk space is made available to users, there is no effective way for the system to prevent multiple users sharing the same UID from possibly interfering with each other in the scratch space. Another disadvantage is the lack of effective accountability. If a user identified with a shared UID misbehaves, it is difficult for the system and site administrator to determine which user is causing the problem, and it is even more difficult to affix blame if the misbehavior is felonious. Finally, with one UID for a set of users, there is no possibility of real persistence for the user on the system. Several mechanisms in UNIX such as message queues and shared memory that provide for persistence across process lifetime will not function well with a shared UID.

The scheme described in this paper is fundamentally different than the PUNCH approach in that it creates a one to one account binding between a user and an account that is identical (as far as the local binding pool is concerned) to a normal local account. There are several advantages to this scheme over the single UID approach. First, this solution is a compromise between requiring one firmly bound account for every user that may use the system (essentially providing a 100% grade of service) versus provisioning only one UID for all users. Additionally, account templates provides a form of persistence to frequent users, but also permits new users to use the system without excessive administrative overhead to manage accounts for new or infrequent users. This approach also allows site administrators to predict and minimize the effects of Grid usage on systems they administer. Thus, the account template approach described in this paper provides a measured form of persistence, identity, and accountability.

The National Partnership for Advanced Computational Infrastructure (NPACI) and the PACI issues individual accounts for every user that is granted an allocation on a host at an NPACI site. The NPACI account creation process uses a database as a centralized repository of users and automating the account creation process using daemons [35].

Globus and PACI issues an X.509 certificate for every user, but a static mapping to a local account using a globusmap file is required at the local host to allows a user to run a job at the site.

3 Account Templates

Using account templates for allocation attempts to take advantage of the potential "locality" of the Grid user's utilization pattern to support hundreds of thousands of Grid users and at the same time provide site administrators an analytical tool for designing and provisioning their systems to limit and meter utilization of their systems by users from the Grid.

The basic idea is to create a pool of "template" accounts that have no permanent association (or "binding") with a user, and to create temporary persistent bindings between template accounts and Grid users. Each template account uses the regular account attributes that are normal for the host system, but the user information refers to a pseudo user, not a real user. For example, for a UNIX host the template account may have the following password file entry:

```
pseudo1:*:12345:23:Psedononymous
User 1:/home/pseudo1:/bin/csh
```

Note that the account has no password – users will be permitted to directly login to the account using an X.509 based ssh [36], and a root level process will be enabled to create a process in the user space of the pseudo user on behalf of a real Grid user after authentication.

Most systems today operate on the implicit assumption that account bindings are permanent. User information, such as the user's real name (GECOS field), home directory, program initialization files (such as .cshrc) are assumed to preexist on the system. For temporary bindings to integrate well with the existing systems, information about the real user's identity needs to be made available to the process and to the system.

3.2 Binding Pools

To address the integration issue with existing systems, there must be some form of administrative domain. A set of 1:1 bindings that associates Grid users to template accounts represents a binding pool. A set of hosts would subscribe to one and only one binding pool. For example, a Linux cluster consisting of 64 hosts would subscribe to one binding pool, which may also be subscribed to by any other collection of hosts. The relationship between hosts and binding pools is shown in Figure 1.

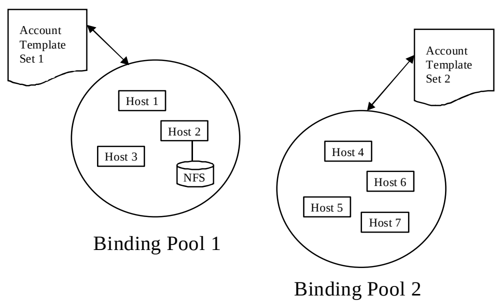


Figure 1: Binding Pools

The purpose of a binding pool is to facilitate use of services on systems that were not designed to operate with temporary account bindings. Local services (such as NFS) that still rely on firm account binding should only be used within the scope of a binding pool. It may be the case that user to account template assignments might be identical across binding pools, but no assumptions should be made about identical assignments across binding pools, since the binding pools are intentionally separate. Higher level distributed services, such as file sharing through AFS, that don't rely on firm account bindings, normally use higher level authentication and authorization mechanisms such as X.509, Kerberos and Akenti to map users to individual account templates within a binding pool domain. The temporary account binding mechanism is natural given that modern distributed systems are moving from reliance on local bindings for authentication and authorization to a greater reliance on global binding and authorization mechanisms.

Account Template States

Firmly bound accounts exists in one of two states: active/available, and unavailable. A firmly bound account is active/available when the account is available for a properly authenticated and authorized use. The firmly bound account in unavailable when the account is still on the system, but has been disabled. Due to the more complex nature of temporary account bindings, additional states are necessary. Each template account in a set of template accounts will be in one of the following states:

- NEW – New local account that has never had any Grid user association. This is different from SCRATCH, which has had an association with a Grid user.
- ACTIVE – Local account with an active, running, online association to a Grid user
- QUIET – Local account with no active online association to a Grid user
- UNAVAILABLE – Local account not available for any associations to a Grid use
- SCRATCH – Persistent assignment of local account to a Grid user has been broken.
- ERROR – An error has occurred in the template, and the template is unavailable for any user.

The following states are transition states between the states described above:

ACTIVE_PENDING
QUIET_PENDING
SCRATCH_PENDING
UNAVAILABLE_PENDING

Valid state transitions for account template states are shown in Figure 2.

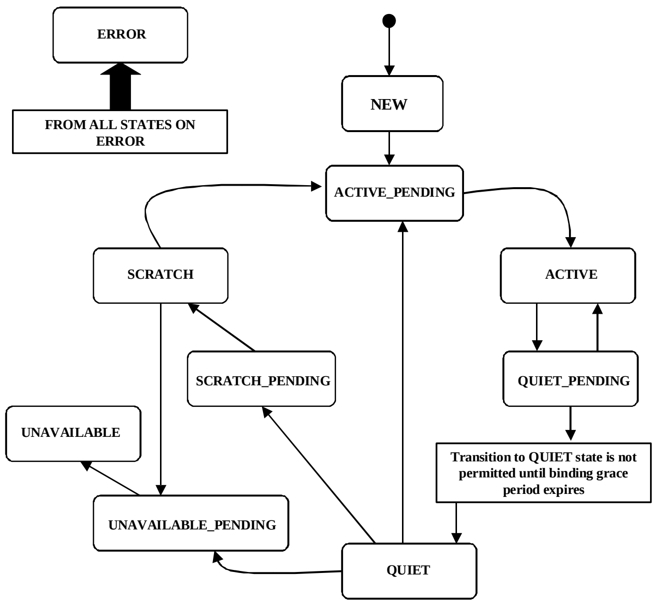


Figure 2: Account Template States

The typical process for assigning a user to an account template from start to finish follows this path is as follows:

1. A resource utilization request arrives in the local domain from the Grid. A valid resource request would include a valid signed X.509 certificate.
2. A process in the local domain (such as the Globus security) checks the X.509 certificate to validate the information contained within it. Once the X.509 certificate has been validated, the user is considered authenticated, but not authorized. The X.509 certificate that is transmitted with the utilization request is retained to permit the security gateway process to perform operations on behalf of the user. This will enable the delegation mechanism that is

described in Security Implications of Typical Grid Computing Scenarios [1].

3. Next, the user resource request is checked against the local authorization mechanism (perhaps Akenti) to determine if the user has sufficient authorization to use the resources they have requested. The Grid resource provider that is considering whether to provide access to the Grid user can verify that the user is a member of a class of users that is authorized to use the resources, and the provider can check the Grid user's originating site to verify that the user has sufficient Grid "money" to pay for the resource use on the local system.
4. At this point, the Grid user is authenticated, and he is authorized to use the resources requested. The next major step is to create an association between the user and a template account.
5. First, it is determined if the user already has an existing QUIET_PENDING template account association. If the user does, then that account template is bound to the Grid account, and the new state for the template is changed through ACTIVE_PENDING to ACTIVE.
6. If the user has no existing template association, the pool of NEW accounts is checked to see if there are any available template accounts. If there are, the new account template is bound to the Grid user and the state of the template is changed through ACTIVE_PENDING to ACTIVE.
7. If there are no available NEW accounts, the SCRATCH pool of accounts is examined to determine if there are any available template accounts. If there are, the account is assigned to the user and the state of the template is changed through ACTIVE_PENDING to ACTIVE.
8. If there are no open SCRATCH accounts, then a QUIET account must be utilized for a template account for the grid user. A replacement policy such as Least Recently Used may be used to move the account from QUIET through SCRATCH_PENDING to the SCRATCH state, where it then may be bound to the Grid user.
9. If there are no QUIET accounts available, and if no QUIET_PENDING grace period timers have expired, then there are no available template accounts that may be bound to the Grid user. An error must be issued to the user and the Grid request is unfulfilled. In practice, this should be a rare occurrence, and the administrators of the Grid resource site must be notified. A Grid resource site should not accept a job from the Grid if it has no available template accounts to bind to a Grid user.
10. Once the binding of a template account to a Grid user is complete, the user's characteristics must be associated with the account, in a fashion similar to Microsoft Windows user profiles. [37]
11. Any registrations that should occur with various Grid services and mechanisms should occur at this point. One example is registering with the Grid Information Service (GIS) for sending and receiving Grid events [3].
12. Binaries and datafiles for the user are then copied into the local host filespace, using access control and permission that were mapped in to the binding pool from the authorization, user credentials and local permissions.
13. The process the Grid user requested is then invoked.
14. Upon completion or termination of the process, the output files are then sent back to the user by some mechanism, and the template moves from ACTIVE through QUIET_PENDING and the grace period timer is set.
15. The resource usage of the process is recorded on the local system and charged against the originating site of the user from the Grid [20].

3 System Issues with Temporary Binding

The majority of systems have been designed with the assumption that firm binding between a user and an account is inviolate. When this assumption is invalidated, several systems issues must be addressed. These issues include persistence, multi-site usage, accounting, and usage metering. This section will examine these issues.

4.1 Dealing with Persistence

On UNIX systems, artifacts may be left upon the completion of a process. These artifacts represent persistent state that the operating system maintains on behalf of the user. Examples of this include files held in the file system, shared memory segments, semaphores and message queues [19]. Any account template solution that creates a temporary binding between users and accounts must also have mechanisms to manage these persistent objects. Additionally, for account tracing purposes, meta-account information about the real user's identity must be retained on the system.

If the system always broke the binding when every job was completed, a lot of unnecessary work would be performed by the system to rebind recently bound accounts. Moreover, the user would need to spend extra effort moving persistent information in and out of the system for every computation. To address these problems, a "grace period" timer is utilized in the state transition from QUIET_PENDING and QUIET states. When a Grid process completes, the binding is not considered breakable until the grace period is expired. This gives the user some guarantee of binding persistence, and allows them to run several consecutive jobs using the bound account template without worrying about losing persistent information that is an artifact of their computation. To allow the Grid resource provider some policy control over this persistence, site administrators must be able set the value of this timer.

4.2 Multi-role Multi-site users

If a Grid user is authorized to use computer resources at several sites on the Grid, then the question arises about the appropriate "usage role" or resource account the computation should be charged against when consuming Grid resources. If the user is consuming resources on behalf of a project that is sponsored at a remote site, then the user's resource consumption should be counted against that project. It would be inappropriate for the Grid user to consume resources from an accounting pool that benefits another project. To allow a Grid user to participate in several projects, a mechanism is needed to link a "real" account at a site with an indirect account reference. Figure 3 illustrates a possible scenario:

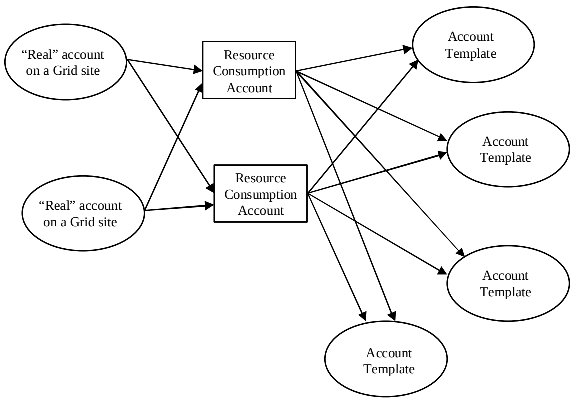

"Real" account on a Grid site
"Real" account on a Grid site
Resource Consumption Account
Resource Consumption Account
Account Template
Account Template
Account Template
Account Template

Figure 3: Multi-User Multi-Account Scenario

The only access a user can have to Grid resources is through the Resource Consumption Account. This requirement enforces a complete accounting of all Grid resource usage. Research is currently underway [21, 39] to define the mechanisms to successfully deal with this situation.

4.3 Authorization and Usage Metering

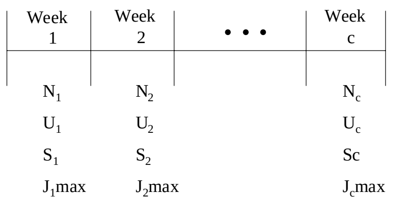
UNIX based systems use a unique UID as the basis for authorization, and as a key for storing and retrieving resource usage information. With the introduction of high level authorization mechanisms, new extensions to UNIX will be required to map artifacts such as X.509 certificates to the local version of authentication. Host based accounting systems will also need to be extended to store a Grid wide unique identifier, such as an X.500 distinguished name (DN) along with the usual accounting information.

In practice, authentication and authorization on hosts for users with template accounts will only work if there is a production quality high level authorization and authentication system in place at the resource site. Providing access to local systems to a pool of potentially hostile users without a secure and trustworthy system in place (such as Kerberos, X.509, and Akenti) to provide site administrators control over who does what on their systems is unwise. Akenti [2] is an example of an X.509 based authorization system that provides fine-grained access control to resource and services.

Usage metering and accounting will still be done locally on the basis of the template account UID, but in terms of metering and accounting for individual users, a global and local accounting system based on X.500 DNs will need to be in place. Work is under way in the Grid community on these mechanisms [21, 20].

5 Determining the Peak Number of Account Templates

For practical considerations, determining an upper bound on the number of template accounts required to satisfy a stream of utilization requests for the Grid is very desirable. Maintaining an arbitrarily large pool of template accounts to satisfy 100% of offered utilization requests would compel a Grid resource provider to incur a large fixed overhead cost, since each template account requires a certain amount of static resources (such as disk space, UID/name space, administrative effort, etc.). To calculate a reasonable upper bound on the number of template accounts necessary to satisfy a stream of job requests, we can use the historical resource utilization of the system as a guide for prediction. Given the historical job arrival rate, job time in the system, and probability of a user arriving at the system based upon aggregate use, it is possible to determine an upper bound on the number of template accounts that will provide a predictable grade of service (G.O.S.) to the offered job stream.
Consider the following:



Where N_i is the number of jobs successfully executed during week$_i$, U_i is the complete list of users (with repeats) that submitted a successful job during the week, S_i is the average time spent in the system for all the jobs that ran that week. During the time period of a week, there will be a period during which the number of jobs in the system will reach a maximum. This peak usage is analogous to the “busy hour” in telephone systems in which the maximum is the daily peak reach in the number of outside telephone lines used to service outgoing telephone calls in a private telephone system [27]. Let the maximum number of jobs in the system during the peak period be represented by $J_i$max.

5.1 Characterizing the Job Stream

To model the characteristics of the offered job stream, some underlying observations of the job stream must be established. Jobs in the job stream are initiated by user actions. Paxson [23] determined that user initiated connections (such as Telnet and FTP) demonstrate exponentially distributed interarrivals and independent arrival events, and could be successfully modeled with a Poisson distribution. Since the job stream is also user initiated, it would be reasonable to hypothesize that the job stream could also be modeled with a Poisson distribution. To verify this hypothesis, the interarrival times between requests in the job stream were analyzed. The interarrival times demonstrated an exponential distribution, and the submission of jobs was assumed to be independent, since each represents a unique job submission event by a user. Based upon these characteristics, and examination of the data, it was determined that the arrival rate of the job stream follows a Poisson distribution. The median of the arrival rate can be calculated from the historical job information, or can be estimated based upon some upper limit of execution in the system, such as a queue time limit.

For a given week$_i$, the distribution of arrival rates into the system follows a Poisson distribution with median λ_i. During the “busy period” of peak utilization of the system, the arrival rate of jobs is much higher than the median arrival rate λ_i. Accurately characterizing the arrival rate during the busy period is complex [27,28, 40], but the arrival rate during the busy period can be approximated by taking advantage of the fact that the arrival rate follows a Poisson distribution. If we use an approximation of the median plus two standard deviations, we should be able to generate a value for the arrival rate that is larger than approximately 98% of the values in the Poisson distribution. Thus, we assume that

$\lambda_i max = \lambda_i + 2s = \lambda_i + 2\surd\ \lambda_i \equiv N_i + 2\surd N_i$

where N_i is normalized to units of hours.
The time spent in the system for all jobs was analyzed and determined to be exponentially distributed (this will be useful later). The average time in spent in the system S_i is the arithmetic mean of the time spent in the system for all the jobs in $week_{i,}$ and thus

$1/\mu_i = S_i$

5.2 Calculating the Peak Number of Jobs

Using Little's Law, the number of simultaneous active sessions during $week_i$ is

$E_i = \lambda_i max/\mu_i$

Where E_i represents the maximum number of session active during the busy hour of the week.

The job stream offered to the system demonstrates the following characteristics:

- There are (potentially) an infinite number of sources
- Job requests arrive at random
- Job requests are serviced in order of arrival
- Refused requests are "lost"
- Time in the system is exponentially distributed

Given that the system has these characteristics, the Erlang-B distribution can be used to predict the probability that an account binding request will be blocked. Moreover, for a desired blocking grade of service (GOS), the minimum number of account templates required can easily be calculated. [26, 27, 28]. The algorithm developed by [26] can be used to calculate the number of template accounts required to satisfy a desired GOS given E_i.

The historical job logs the for the IBM SP-2 systems at the University of Michigan Center for Parallel Computing and the Advanced Computing Center for Engineering & Science (ACCES) system at the University of Texas at Austin were analyzed to measure the ability of this method to successfully predict the maximum number of account templates required to satisfy an offered job stream. The University of Michigan logs contain 7,294 jobs over a period of 44 weeks. The University of Texas logs contain 3,160 jobs over a period of 27 weeks. Jobs that were in the system less than 10 minutes were filtered out of the job stream, to remove jobs that did not actually execute in the system. From the logs, the average arrival rate for all of the weeks was calculated:

U-M
$\lambda = (\Sigma_C \lambda_i) /\ c = 0.6706$ users / hour
$\lambda max = \lambda + 2\ SQRT(\lambda) = 2.308$ users/hour

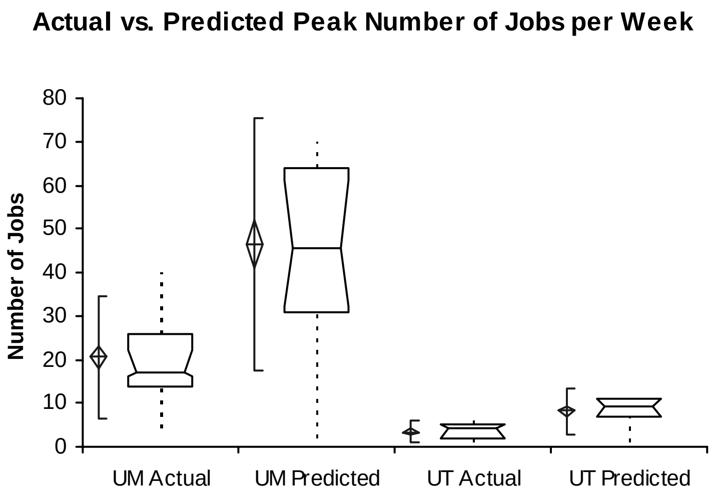
U-T
$\lambda = (\Sigma_C \lambda_i) /\ c = 0.5245$ users / hour
$\lambda max = \lambda + 2\ SQRT(\lambda) = 1.973$ users/hour

The average time in the system over all the weeks was also calculated:
U-M: $1/\mu = 18.86$ hours
U-T: $1/\mu = 11.588$ hours

Thus,
U-M: $E = \lambda max/\mu\ = 43.53$ sessions
U-T: $E = \lambda max/\mu\ = 22.86$ sessions

If we then calculate the peak number of sessions with ½% GOS using the Erlang-B Loss Formula [22, p. 273], Emax = 59 sessions for U-M and Emax=36 for U-T.

To measure the ability of this method to utilize information from preceding weeks to predict the peak number of jobs and unique users for a week, the logs from both U-M and U-T were used to successively predict these values. This approach is called the *moving averages approach* in [27] and is used for forecasting demand in telecommunications circuits. After the first few weeks, the technique was fairly successful in predicting appropriate values to satisfy all of the requests for that week measured from the historical logs. Figure 4 shows that the peak number of jobs calculated compared with the actual peak number of jobs measured per week. The predicted peak value of 59 exceeds all of the measured values, and should be able to provide a ½% GOS over all weeks.



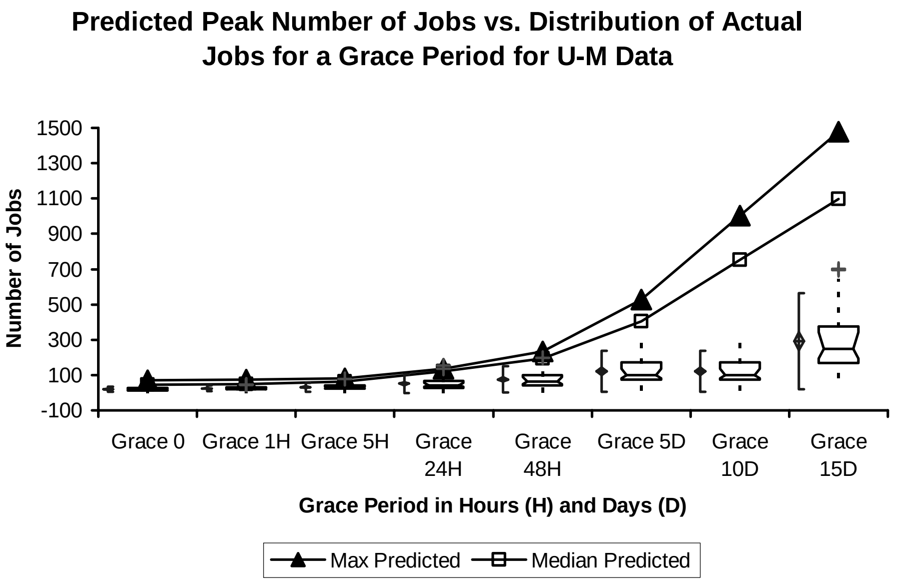
	Median	IQR	95% CI of Median
UM Actual	17	11.750	16.000 to 22.000
UM Predicted	45.5	33.000	32.000 to 61.000
UT Actual	4	3.000	2.000 to 5.000
UT Predicted	9	4.000	7.000 to 11.000

Figure 4: Peak vs. Predicted Number of Jobs per Week

The data table in Figure 4 contains the median, interquartile range, and the 95% confidence interval of the median for the actual and predicted peak number of jobs per week for U-M and U-T. The ability to calculate the predicted peak number of jobs based upon the average arrival rate and time in the system is useful for allowing system managers to provision the system for a known peak number of jobs.

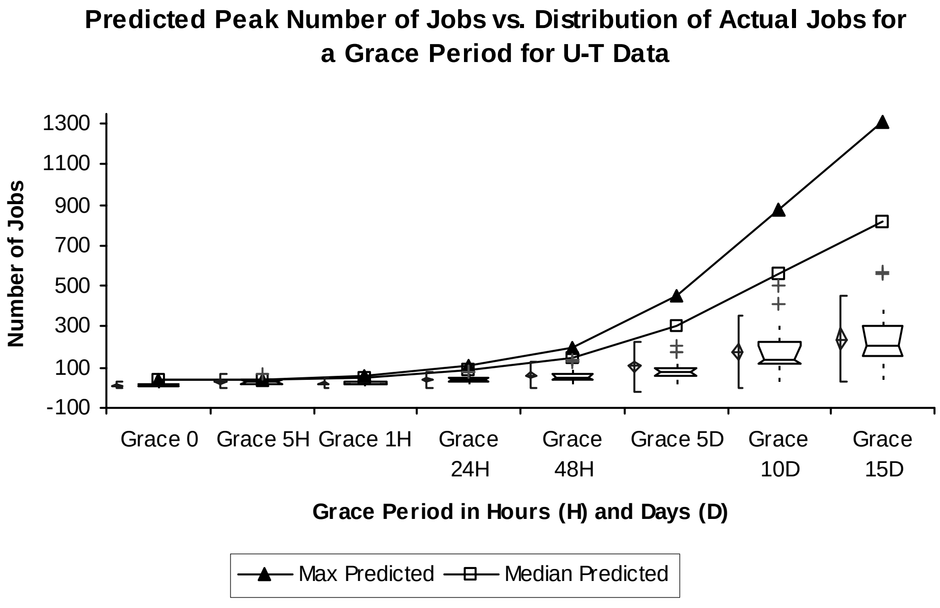
5.3 Adding a Grace Period to the Binding

The results presented in the previous section is for the scenario in which jobs arrive, are temporarily bound to an account template, and the binding between the user and the template account lasts only as long as the job is in the system. In practice, however, immediately breaking the binding is undesirable for the user and for the system. If a user frequently returns to the system, the aggregate overhead of setting up and tearing down account bindings would be excessive. From the user's point of view, it would be desirable to have a period of time (a "grace period") after the completion of the jobs to be able to collect or analyze the output of the execution. If we introduce a grace period of any significant length to the system, however, some changes must be made to the predictive model to take into account the effects of the user revisiting the account, which in effect extends the time the account binding is in the system. If we introduce a uniform grace period G that is selected by the systems manager, the average holding time for each job $1/\mu$ will be extended to $(1/\mu) + G$, and the corresponding weekly $1/\mu_i$ and aggregate $1/\mu$ will also be extended by G. The arrival rate of the offered jobs to the system λ_i would remain unchanged, and the peak predicted number of jobs would also be scaled appropriately for G. To verify this result, the University of Michigan and University of Texas logs were analyzed to determine the peak number of jobs in the system over all the weeks in the logs for a given grace period G. Figure 5 show the predicted vs. actual peak number of jobs for a set of grace periods from 0 to 15 days for University of Michigan and University of Texas.



	n	Max Predicted	Median Predicted	Median Actual	IQR Actual	95% CI of Median Actual
Grace 0	44	69	46	17	11.750	16.000 to 22.000
Grace 1H	44	73	49	22	10.250	20.000 to 26.000
Grace 5H	44	83	62	27.5	21.250	23.000 to 33.000
Grace 24H	44	136	123	43	40.250	36.000 to 56.000
Grace 48H	44	235	194	62	58.500	48.000 to 87.000
Grace 5D	44	526	405	100.5	96.750	81.000 to 136.000
Grace 10D	44	1004	753	100.5	96.750	81.000 to 136.000
Grace 15D	44	1480	1097	247	205.500	193.000 to 341.000

Figure 5.1: Predicted Peak Number of Jobs vs. Actual Peak Number of Jobs for U-M



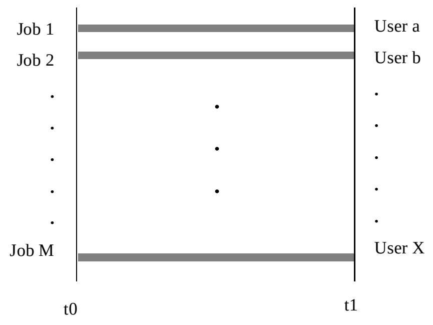
	n	Max Predicted	Median Predicted	Median Actual	IQR	95% CI of Median Actual
Grace 0	27	39	36	11	8.500	8.000 to 16.000
Grace 5H	27	42	38	25	16.000	20.000 to 34.000
Grace 1H	27	54	48	20	9.000	15.000 to 23.000
Grace 24H	27	109	92	34	19.000	28.000 to 44.000
Grace 48H	27	194	147	49	34.000	39.000 to 70.000
Grace 5D	27	455	303	80	40.000	62.000 to 101.000
Grace 10D	27	878	558	138	106.500	121.000 to 208.000
Grace 15D	27	1309	814	203	149.500	161.000 to 307.000

Figure 5.2: Predicted Peak Number of Jobs vs. Actual Peak Number of Jobs for U-T

This analysis demonstrates that the model remains valid when an additional grace period is introduced.

5.4 Predicting the Number of Unique Users in the Job Stream

At the busy period during the week, if the actual peak number of jobs is M, the situation is as follows:



Assuming that jobs start at t0 and complete at or after t1. For the worst case, one could assume that each job is assigned to a unique individual user, and thus would require M template accounts. In practice, however, the number of jobs attributable to a user is the product of the probability of a job originating from that user in the job stream and M. This is due to the theorem that a Poisson process can be partitioned into a set of impendent Poisson processes [22, p. 74]. To apply this theorem, we must determine the probability of the user being the originator of a job in the job stream from the historical logs.

Let p_i = Probability (user$_i$) = (Number of jobs for user$_i$ / Total Number of Jobs). Thus,

$\lambda_{avg} = p_1 \lambda_{avg} + p_2 \lambda_{avg} + p_3 \lambda_{avg} + \ldots + p_k \lambda_{avg}$

$\lambda_{max} = p_1 \lambda_{max} + p_2 \lambda_{max} + p_3 \lambda_{max} + \ldots + p_k \lambda_{max}$

where k is the number of users in the historical logs. Thus,

(1) $E_{max} = 1/\mu \, (p_1 \lambda_{max} + p_2 \lambda_{max} + p_3 \lambda_{max} + \ldots + p_k \lambda_{max})$

and

$E_{u1} = (p_1\lambda_{max})/\mu$, $E_{u2} = (p_2\lambda_{max})/\mu$, $E_{u3} = (p_3\lambda_{max})/\mu$, ..., $E_{uk} = (p_k\lambda_{max})/\mu$

The predicted number of unique users at the busy period will be the number of terms in (1) that have a values greater than 1.

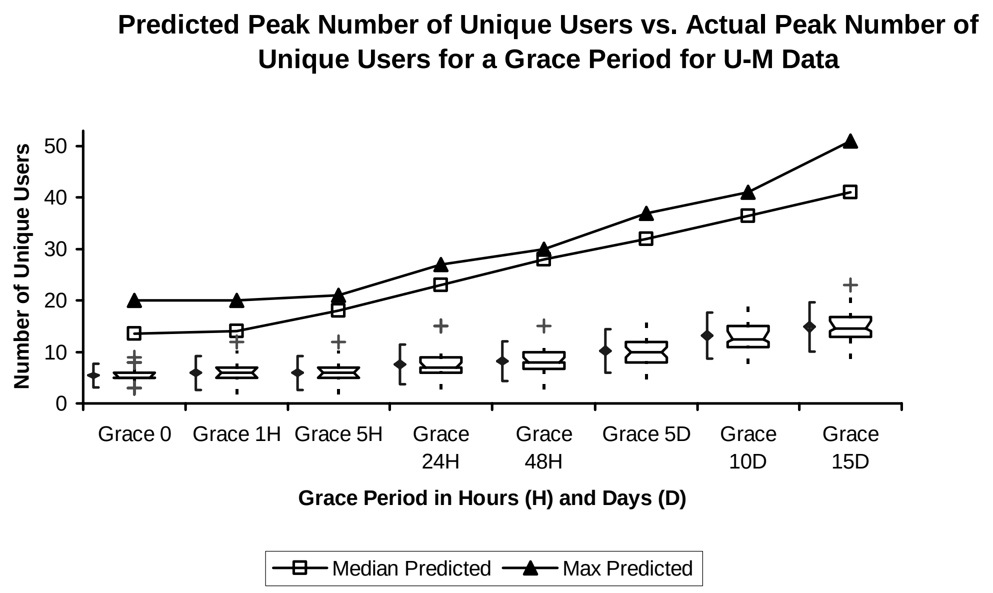
(2) $\| U \|$ where $U = \{ E_{ui} : E_{ui} > 1\}$

We can then add the grace period G described in the previous section to (1):

$E_{max} = (1/\mu + G) \, (p_1 \lambda_{max} + p_2 \lambda_{max} + p_3 \lambda_{max} + \ldots + p_k \lambda_{max})$

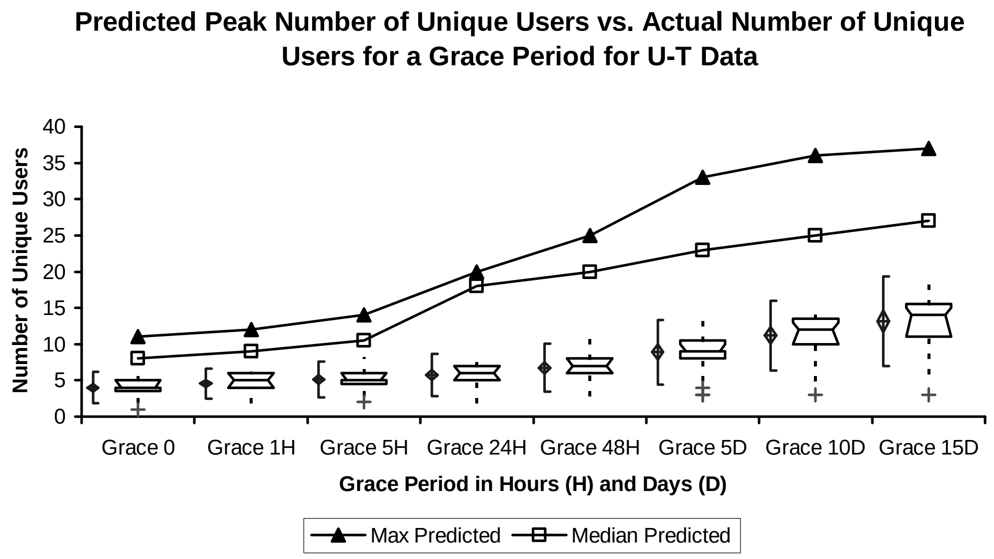
The period of time window used in the historical job log to calculate the probability for each user that is used to calculate E_{max} must be at least 2 $(1/\mu + G)$ to capture all possible jobs in the window. For the most accurate results p_i should be based upon an analysis of the job log as far back into the past as is practical.

Figure 7 shows the predicted peak number of individual users per week vs. the actual peak number of individual users for U-M and U-T.



	n	Max Predicted	Mean Predicted	Median Actual	IQR	95% CI of Median Actual
Grace 0	44	20	14	5.000	1.000	5.000 to 6.000
Grace 1H	44	20	14	6.000	2.000	5.000 to 7.000
Grace 5H	44	21	18	6.000	2.000	5.000 to 7.000
Grace 24H	44	27	23	7.000	3.000	7.000 to 8.000
Grace 48H	44	30	28	8.000	3.250	8.000 to 9.000
Grace 5D	44	37	32	10.000	4.000	9.000 to 11.000
Grace 10D	44	41	37	12.500	4.000	12.000 to 14.000
Grace 15D	44	51	41	14.500	3.750	14.000 to 16.000

Figure 6.1: Predicted Peak Number of Users vs. Actual Peak Number of Users for U-M



	n	Max Predicted	Median Predicted	Median Actual	IQR	95% CI of Median Actual
Grace 0	27	11	8	4	1.500	4.000 to 5.000
Grace 1H	27	12	9	5	2.000	4.000 to 6.000
Grace 5H	27	14	11	5	1.500	5.000 to 6.000
Grace 24H	27	20	18	6	2.000	5.000 to 7.000
Grace 5D	27	25	20	7	2.000	6.000 to 8.000
Grace 48H	27	33	23	9	2.500	9.000 to 10.000
Grace 10D	27	36	25	12	3.500	11.000 to 13.000
Grace 15D	27	37	27	14	4.500	11.000 to 15.000

Figure 6.2: Predicted Peak Number of Users vs. Actual Peak Number of Users for U-T

It can be demonstrated that as the grace period G approaches infinity, the maximum number of users calculated from the probability vector approaches the number of individual users that utilized the system. This corresponds to the firm binding case, where over a very long period of time, the number of users that use the system matches the number of users contained within the password file on the system.
With the addition of the probability of individual users using the system, system managers can then confidently provision the system for a known peak number of users based upon the historical use information of the system.

5.5 Application: Using Theory to Determine Grace Period and Number of Template Accounts

Now that we can accurately predict an upper bound on the number of template accounts and individual users a Grid resource site would service given the historical logs, GOS, and grace period, we now want to be able to easily calculate the number of template accounts a site would need to support a GOS and grace period selected by the site administrator. The process to do this is as follows: first, calculate average holding time $1/\mu$ by calculating the arithmetic mean of the time in hours in the system for all jobs over a certain filter threshold (to filter out unsuccessful jobs); second, calculate arrival rate λ by dividing the total number of filtered jobs in the log by the number of hours the log covers; third, calculate the user probability vector **p** by taking the inverse of the number of times a jos comes from a particular user (for example, if user X has 10 jobs, Prob(X) = 1/10). Now, calculate $\lambda\text{max} = \lambda + 2*\text{SQRT}(\lambda)$, and calculate $E = \lambda\text{max}/\mu$. If you have a package to calculate the Erlang Loss Formula, such as Qsim for Excel [25], or a Java applet [26], you can use E and a selected GOS as inputs to calculate E_{max}, and calculate the peak number of users by the number of elements of the vector $\mathbf{u} = \mathbf{p}E_{max}$ that are greater than one. This will give you the peak number of users that will utilize the system with a known probability of blocking a job due to a depletion of template accounts.

For example, from the University of Michigan data, $\lambda\text{max} = 2.308$ users/hour, $1/\mu = 18.86$ hours, and Emax = 43.53. With user probability vector **p** calculated from the logs, and a desired GOS of 20% blocked jobs, U-M should provision 9 template accounts. If the GOS is increased to 1% blocked jobs, the required number of template accounts rises to 15 accounts. Since the difference between 9 and 15 accounts is small, U-M may decide that the small additional cost of providing 6 accounts to greatly improve the GOS is justified. If a grace period of 40 days is added to $1/\mu$ for a value of 978.86 hours, Emax now has the value of 2339 and using **p** the peak number of template accounts required to support a GOS of 1% is 61.

5. 6 Summary

In this section, it was demonstrated that historical log information maintained on the system can be used to quickly and simply calculate the peak number of jobs and users that may utilize the system. As Grid technology progresses and the mix of jobs offered to Grid systems comes more frequently from automated processes, the self-similar arrival processes described by Paxson [23] will become a more significant factor and should be taken into account for future models. Another area of work is the prediction of peak job and users for jointly scheduled systems, such as the Berkeley Millennium [24] system, Globus, and the U-M/U-T NPACI Grid.

6 Application : The Visible Human Project

The Visible Human Project (VHP) at the University of Michigan sponsored by the National Institutes of Health [31, 41] is attempting to create a Next Generation Internet production system to serve v
isible human datasets to learning systems at medical training facilities throughout the United States. The VHP has developed an anatomical dataset navigation tool called Edgewarp [32, 33] that will require significant high performance computation and storage resources to render and deliver real-time flythrough images of anatomic data sets under haptic controls. Design parameters for the VHP requires support for at least 40 simultaneous users in a teaching clinic at a site for Nursing, Medical and Dental students.

For the VHP to be able to successfully support a large computational load with a transient pool of users, the account template mechanism is critical to allow the VHP software to convey a student's X.509 certificate from an entity such as a smart card to a Grid resource site that is willing to provide computational or storage resources for the student's use of the system. If each Grid resource provider is required to establish an account for each student before the term, and then remove the account at the end of the term, the administrative overhead and costs associated with this activity would be so large that very few sites would be able to participate.

BRIAN – ADD MORE HERE.

7 Conclusion and Future Work

In this paper, an alternative to existing account allocation techniques was presented that addressed the problems of identity, persistence and accountability present in these systems. A practical technique for predicting an upper bound on the number of jobs and users that is based on historical use was presented. Analysis of this technique using log data from two different resource sites was presented that demonstrated the success of the technique. Finally, application of the account allocation mechanism presented in this paper was described.

For future work, investigation of the effect of self-similar arrival processes on this model should be done. Another area for future work is an thorough investigation of the effects of temporary account bindings on Operating Systems design, and investigation of what changes would be required on a UNIX system to support temporary account bindings.

REFERENCES

[1] Humphrey, Marty and Thompson, Mary. Security Implications of Typical Grid Computing Usage Scenarios. Grid Forum Security Working Group Draft. October 2000.

[2] Thompson, M., Johnston, W., Mudumbai, S., Hoo, G., Jackson, K., Essiari, A. Certificate-based Access Control for Widely Distributed Resources. Proceedings of the Eighth Usenix Security Symposium, Aug. `99.

[3] Smith, W., Gunter, D. A Grid Information Service Schema for Grid Events. Grid Forum GP working group draft. October 2000.

[4] Dyer, Stephen P., "The Hesiod Name Server," USENIX Technical Conference, Dallas, Texas, Winter 1988.

[5] Steiner, Neuman, Schiller, Kerberos: An Authentication Service for Open Network Systems, USENIX Technical Conference, Dallas, Texas, Winter 1988.

[6] R. Housley, W. Ford, W. Polk, D. RFC 2459 Internet X.509 Public Key Infrastructure Certificate and CRL Profile. January 1999

[10] D. H. J Epema, Miron Livny, R. van Dantzig, X. Evers, and Jim Pruyne, "A Worldwide Flock of Condors : Load Sharing among Workstation Clusters"
Journal on Future Generations of Computer Systems Volume 12, 1996

[11] The Network Desktop of the Purdue University Network Computing Hubs. Nirav H. Kapadia and
Jose' A. B. Fortes. Technical Report TR-ECE-99-1, School of Electrical and Computer Engineering, Purdue University. January 1999.

[12] PUNCH: An Architecture for Web-Enabled Wide-Area Network-Computing. Nirav H. Kapadia
and Jose' A. B. Fortes. Cluster Computing: The Journal of Networks, Software Tools and Applications; special issue on High Performance Distributed Computing. September 1999.

[13] Doster , B., Rees, J. Third-Party Authentication for the Institutional File System. University of Michigan Center for Information Technology Integration Technical Report 92-1. http://www.citi.umich.edu/techreports/citi-tr-92-1.ps.gz.

[14] AFS for Windows. Transarc Corporation. http://www.transarc.ibm.com/Library/documentation/afs/3.5.windows.

[15] Howard, J. H., Kazar, M. L., et. al. “Scale and Performance in a Distributed File System,” ACM Transactions on Computer Systems, 6(1):51-81, 1988.

[16] Kazar, Michael Lean, Leverett et al., “Decorum File Systems Architectural Overview,” USENIX Conference Proceedings, USENIX Association, Berkeley, CA, Anaheim June 1990.

[17] Nelson, M. L.; Priest, T. L., and Bianco, D. J.: “Experiences with DCE/DFS in a Production Workstation Cluster Environment,” NASA TM (in preparation) working version at: http://www.larc.nasa.gov/~mln/dfs/dfs.html.

[18] Foster, I. and Kesselman, C. Globus: A Metacomputing Infrastructure Toolkit, International Journal of Supercomputing Applications, 11(2): 115-128, 1997. http://www.globus.org.

[19] UNIX System Laboratories. STREAMS Modules and Drivers, Prentice-Hall, 1992.

[20] Hacker, T. Thigpen, W. Distributed Accounting on the Grid. Grid Forum Working Draft.

[21] Buyya, R., Abramson, D., Giddy, J. “An Economy Grid Architecture for Service-Oriented Grid Computing”. Pre-publication work in progress.

[22] Wolff, R. Stochastic Modeling and the Theory of Queues. Prentice Hall, 1989.

[23] V. Paxson and S. Floyd, "Wide-area Traffic: The Failure of Poisson Modeling," IEEE/ACM Transactions on Networking, pp.226-244, June 1995.

[24] University of California, Berkeley. UC Berkeley Millenium Project, 1997. http://www.millennium.berkeley.edu.

[25] 3 Point Technologies, Inc. Qsim Modeling Functions for Excel. http://www.3ptech.com/qsim.

[26] S. Qiao and L. Qiao, A Robust and Efficient Algorithm for Evaluating Erlang B Formula, Technical Report CAS98-03, Department of Computing and Software, McMaster University, Ontario, Canada, L8S 4L7, August 1998.http://www.cas.mcmaster.ca/~qiao/publications/erlang/newerlang.html

[27] Green, James H. The Irwin Handbook of Telecommunications Management. Irwin Professional Publishing, 1996.

[28] Intel Support Document #8150. A Traffic Engineering Model for LAN Video Conferencing. http://support.intel.com/support/proshare/8150.htm.

[29] Foster, I., and Kesselman, C. (editors), The Grid: Blueprint for a New Computing Infrastructure, Morgan Kauffman Publishers, 1999.

[30] William E. Johnson, Dennis Gannon, and Bill Nitzberg. Grids as production computing environments: The engineering aspects of NASA's information power grid. In Eighth IEEE International Symposium on High Performance Distributed Computing. IEEE, August 1999.

[31] University of Michigan Visible Human Project. http://vhp.med.umich.edu.

[32] Bookstein, F.L. and Green, W.D.K. Edgewarp 3D: A Preliminary Manual. Posted to the Internet as ftp://brainmap.med.umich.edu/ pub/edgewarp3.1/manual.html, 1998.

[33] B. D. Athey, A. W. Wetzel, and W. D. K. Green. Navigating solid medical images by pencils of sectioning planes. Pp. 63--76 in Mathematical Modeling, Estimation, and Imaging, eds. D. Wilson, H. Tagare, F. Bookstein, F. Preteaux, and E. Dougherty, Proc. SPIE, vol. 4121, 2000.

[34] Schopf, J, Nitzberg, B. Grid: The Top Ten Questions. Northwestern University CS Technical Reports #CS-00-05.

[35] Personal Communications with Victor Hazlewood, HPC Systems Manger, San Diego Supercomputer Center. http://www.sdsc.edu/~victor.

[36] Foster, I., Kesselman, C., Tsudik, G., Tuecke, S. “A Security Architecture for Computational Grids.” Proceedings of the Fifth ACM Conference on Computer and Communication Security.

[37] Dobbins, G. ND_GINA: An Alternative Authentication Method from Windows NT. Technical Report. http://www.nd.edu/~dobbins/ntarch/nd_gina_doc.html

[39] Scott Jackson, QBank: A Resource Management Package for Parallel Computers. Pacific Northwest National Laboratory, USA, 2000.

[40] Alanyali, M. and Hajek, B. (1996), “On Load Balancing in Erlang Networks”, Stochastic Networks: Theory and Applications, F. P. Kelly, S. Zachary, and I. Ziedens (Eds.), Oxford University Press.

[41] M. J. Ackerman, “The Visible Human Project,” J. Biocomm., vol. 18, p 14, 1991.